CHAPTER – SIX

CODING

6.1 Overview

This section of the SRS includes the actual programming scripts used in the project.

6.2 Dataprep.R

```
library("csv")

# FOR TRAINING DATASET

df = read.csv(file.choose(), header = T)

df = df[,!apply(is.na(df), 2, all)]

# having a look at first set of values in data frame

head(df)

# taking backup of original dataset

dfb <- df

# structure of the datframe

str(dfb)

# summary and structure of the first variable: Gender ---------------------------------------

summary(dfb$Gender)

str(dfb$Gender)

# getting the indices of the rows where data is missing in Gender column

toberemoved <- which(dfb$Gender=="")

k <- dfb[dfb$Gender=="",]    # k is a dataframe with 13 observations

class(toberemoved)   # integer

length(toberemoved)  # 13



# remomving the rows
```

```
dfb <- dfb[-toberemoved,]
# reconfiguring coulmn
dfb$Gender <- as.character(dfb$Gender)
dfb$Gender <- as.factor(dfb$Gender)
# summary and structure of the second variable: Married -----------------------------------
summary(dfb$Married)
str(dfb$Married) # factor variable
 # getting the indices of the rows where data is missing in Married column
toberemoved <- which(dfb$Married=="")
k <- dfb[dfb$Married=="",] # k is dataframe with three observations
class(toberemoved)  # integer
length(toberemoved)  # 3
# removing the rows with blank Married status
dfb <- dfb[-toberemoved,]
# reconfiguring columns
dfb$Married <- as.character(dfb$Married)
dfb$Married <- as.factor(dfb$Married)
```

6.3 EDA.R

```
# bar plot for gender
countgender <- table(dftrain$Gender)
countgender
gendernames <- dimnames(countgender)
gendernames
par(mar = c(1.5,1.5,1.5,1.5)+3)
barplot(countgender, names.arg = gendernames[[1]], xlab = "Gender",
          ylab = "Number", ylim = c(0,250),las = 1,
          cex.names = 0.7)
```

```
fcountg <- as.data.frame(countgender)
fcountg

library(ggplot2)
genderplot <- ggplot(fcountg, aes(Var1, Freq)) + geom_bar(stat = "identity",
                         width = 0.5, fill ="steelblue") +
                         theme(plot.margin = margin(2,2,2,2,"cm")) +
                         labs(title = "plot (gender)", x = "Gender", y = "Count")
genderplot



# bar plot for married
countmarried <- table(dftrain$Married)
as.data.frame(countmarried)

marriedplot <- ggplot(as.data.frame(countmarried), aes(Var1, Freq)) +
                         geom_bar(stat = "identity",width = 0.5, fill ="steelblue") +
                         theme(plot.margin = margin(2,2,2,2,"cm")) +
                         labs(title = "plot (Married)", x = "Married", y = "Count")
marriedplot
# principle component Analysis
pcaResult <- prcomp(dfnum[,c(2:13)])
pcaResult$rotation
# correalation matrix
str(dfnum[,c(2:13)])
cormatelements <- dfnum[,c(2:13)]
str(cormatelements)
cormat <- cor(cormatelements, use = 'everything', method = "pearson")
```

```
cormatround <- round(cormat,2)

cormatround

# reordering the correlation matrix elements

cormatround <- reorder_correlation_matrix(cormatround)

cormatround_upper <- get_upper_tri(cormatround)

cormatround_upper


# melting the cormat round upper

library(reshape2)

melted_cormatround_upper <- melt(cormatround_upper, na.rm = T)

melted_cormatround


correlation_plot <- ggplot(data = melted_cormatround_upper, aes(Var2, Var1,
                              fill =    value)) +
                               geom_tile(color = "white") +
                                scale_fill_gradient2(low = "blue", high = "red",
                                mid = "white",
                                midpoint = 0, limit = c(-1,1), space = "Lab",
                                name="Pearson\nCorrelation") + theme_minimal() +
                               theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                size = 10, hjust = 1)) + coord_fixed()
```

```
correlation_plot  +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    axis.ticks = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.6, 0.7),
    legend.direction = "horizontal")+
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
                               title.position = "top", title.hjust = 0.5))
```

6.4 ModelPrep.R

```
# partitioning Train : Valid :: 60 : 40
partidx <- sample(1:nrow(dftrain), 0.6*nrow(dftrain), replace = F)
dftrain <- dftb[partidx,]
dfvalid <- dftb[-partidx,]

dftrain <- dftrain[,-c(1)]
dfvalid <- dfvalid[,-c(1)]

############################################################ Decision Tree
library(rpart)
```

```
library(rpart.plot)

# rel error is the ratio of incorrectly classified traiining records
# after doing a split to incorrectly classified training records
# at the root node (naive  rule)

# xval (default value = 10)
# pruning using rpart's prune
mod1 <- rpart(Loan_Status ~ ., method = "class", data = dftrain,
          control = rpart.control(cp = 0, minsplit = 2, minbucket = 1,
                                  maxcomplete = 0, maxsurrogate = 0,
                                  xval = 10)
          )
pmod <- prune(mod1, cp = cp1)
prp(pmod)



# performance on training partition
bmodtr <- predict(pmod, dftrain, type = "class")
#classification accuracy #0.79
mean(bmodtr == dftrain$Loan_Status)
# misclassification error #0.20
mean(bmodtr != dftrain$Loan_Status)

# performance on validation partition
bmodvr <- predict(pmod, dfvalid, type = 'class')
#classification accuracy #0.82
mean(bmodvr == dfvalid$Loan_Status)
# misclassification error #0.17
```

```
mean(bmodvr != dfvalid$Loan_Status)
################################################ Logistic regression model
mod2 <- glm(Loan_Status ~ ., family = binomial(link = "logit"),
            data = dftrain)
summary(mod2)

# on training partition
lrmodrt <- predict(mod2, dftrain, type = "response")
lrmodrt <- ifelse(lrmodrt > 0.5,"Y","N")

#classification accuracy #0.80
mean(lrmodrt == dftrain$Loan_Status)
# misclassification error #0.19
mean(lrmodrt != dftrain$Loan_Status)

# on validation partition
lrmodrv <- predict(mod2, dfvalid, type = "response")
lrmodrv <- ifelse(lrmodrv > 0.5,"Y","N")

#classification accuracy #0.79
mean(lrmodrv == dfvalid$Loan_Status)
# misclassification error #0.20
mean(lrmodrv != dfvalid$Loan_Status)
```

6.5 Prediction on User defined Values.R

```
# enter Values Here
Gender <- "Male" # "Male"  or "Female"
Married <- "Yes" # "Yes" or "No"
```

```
Dependents <- 1 #  "0", "1", "2" or "3+"
Education <- "Graduate" # "Graduate" or "Not Graduate"
Self_Employed <- "No" # "Yes" or "No"
Applicant_Income <- 4853 # [150, 81000]
Coapplicant_Income <- 1580 # [0, 33837]
Loan_Amount <- 128 # [9, 600]
Loan_Amount_Term <- 360 #[36, 480]
Credit_History <- 1 # {0, 1}
Property_Area <- "Rural" #  "Rural", "Semiurban" or "Urban"
# prediction from decison tree model
dtpr <- predict(pmod, pre_Defined_Values[nrow(dfn),], type = "class")
dtpr
ifelse(dtpr == "Y", "Loan will be Passed !", "Loan Will not be Passed !")
# prediction from logistic Regression model
logRegpr <- predict(mod2, pre_Defined_Values[nrow(dfn)], type = "response")
ifelse(logRegpr > 0.5, "Loan will be Passed !", "Loan Will not be Passed !")
```